EXHIBIT 21

Sysman: A Virtual File System for Managing Clusters

Mohammad Banikazemi, David Daly, and Bulent Abali - IBM T. J. Watson Research Center

Pp. 167–174 of the *Proceedings of the 22nd Large Installation System Administration Conference (LISA '08)* (San Diego, CA: USENIX Association, November 9–14, 2008).

Abstract

Sysman is a system management infrastructure for clusters and data centers similar to the /proc file system. It provides a familiar yet powerful interface for the management of servers, storage systems, and other devices. In the Sysman virtual file system each managed entity (e.g., power on/off button of a server, CPU utilization of a server, a LUN on a storage device), is represented by a file. Reading from Sysman files obtains active information from devices being managed by Sysman. Writing to Sysman files initiates tasks such as turning on/off server blades, discovering new devices, and changing the boot order of a blade. The combination of the file access semantics and existing UNIX utilities such as grep and find that operate on multiple files allow the creation of very short but powerful system management procedures for large clusters. Sysman is an extensible framework and has a simple interface through which new system management procedures can be easily added. We show that by using a few lines of Linux commands system management operations can be issued to more than 2000 servers in one second and the results can be collected at a rate of more than seven servers per second. We have been using Sysman (and its earlier implementations) in a cluster of Intel and PowerPC blade servers containing hundreds of blades with various software configurations.

Introduction

Managing clusters made of hundreds and thousands of servers with various types of storage and network devices consumes a significant amount of system administration resources. Heterogeneity of devices in a cluster or data center and a plethora of command line interfaces, graphical user interfaces (GUI), and web based solutions available for managing different device types have made the management of these systems an ever more challenging task. Ethnographic studies of system administrators have shown deficiencies in current system management tools, including 1) poor situational awareness from having to interface with several management tools, 2) lack of support for planning and rehearsing complex system management procedures, and 3) incomplete functionality requiring system administrators to build their own tools [14]. It is also shown that the ability to automate and script system management functionality is crucial [14].

Currently, there are several monitoring and management solutions for performing system administrative tasks in data centers and clusters [5, 8, 10, 12, 13, 20, 22, 23, 26]. Each of these solutions use a different graphical or web based interface. These solutions vary in how radical they are and how different an approach they require for dealing system administration and monitoring tasks. We discuss several of these solutions in the

Related Work section.

In our work we use a simple interface through which complicated system administrative tasks can be automated easily. We aim to design and provide a powerful yet familiar interface on top of which various system administrative tools can be built. In order to make such an interface successful, instead of providing all the commands for performing various tasks on all devices (which is impossible), we design the system such that manufacturers (and developers) can add the required programs and scripts for a given device to the system. We also design the system such that we take advantage of a large set of familiar system administration utilities (such as filesystem utilities that can be easily used for manipulating large number of files). We call our system Sysman.

Sysman is an easily extensible framework. Our work was initiated by the need to manage a large set of IBM BladeCenters containing tens and hundreds of IBM blades in our lab. In our first implementation, we provided agents mainly targeting IBM BladeCenter [19] platform management and basic Linux server management. Our BladeCenter platform management agents leveraged the libraries developed for [16]. However, we designed Sysman such that it can be enhanced by additional scripts and executables provided by the users and third parties, using a simple and well defined interface. No change to the Sysman file system itself is required for such additions.

Sysman provides a unified and simple interface for managing various devices found in clusters and data centers. Sysman provides a /proc-like interface. The Sysman file system is layered over the command line interfaces and other tools available for managing devices, and provides a very simple yet powerful interface for accessing all these tools in a unified manner. The file system semantics enables the use of for example existing UNIX commands such as find, sort, and grep, and script programming to manage large clusters of servers as easily as a single server. Considering that UNIX (and Linux) file system commands are very simple yet can be utilized to perform very complicated tasks operating on many files, Sysman provides a very powerful and easy to use interface for cluster management. Furthermore, since most system administrators are already familiar with UNIX file system commands and writing shell scripts, they will have a very short learning curve with Sysman. Figure 1 contains three examples of how simply Sysman can be used to monitor and/or control a large number of devices.

```
### A simple command for finding all servers which are turned on
find /sysman/systems -name power | xargs grep -l on
### List all the blade chassis whose temperature is above 35 degrees
find . -name 'temperature_mp' | xargs grep "" | sed s/:/" "/g | \
        awk '{ if ($2 >= 35.0) print $1 "temperature is: " $2}'
### A simple script to collect the vmstat of all servers in background
#!/bin/bash
LIST=`find /sysman/systems -name command`
for i in $LIST; do
    echo "vmstat " > $i &
done
```

Figure 1: Examples of Sysman usage.

In addition to collecting data from servers in a cluster, Sysman uses exactly the same interface to monitor and manage various types of devices. This provides an opportunity to integrate various system management domain into one and reduce the system management cost. For a device to be Sysman enabled, it is enough to have a command line interface through which the device can be monitored and managed. Once such an interface exist, simple scripts can be added to the system to provide the basic methods for managing these devices. Furthermore, as use of virtual machines become more widely used, Sysman can be utilized to create, monitor, manage, and destroy them as well.

System management tasks are performed by accessing Sysman files. In particular, all tasks are accomplished by reading or writing a file or a directory look up operation. Sysman is a virtual file system similar to the Linux /proc or /sys file systems. Sysman creates a virtual file system where accessing files in the /sysman directory results in execution of related system management tasks. Sysman virtual file system is typically created on a management server. Each device or system is represented as a directory. Each device (or system) directory contains files through which various components of the device can be managed. Sysman files are not only bytes on disk but contain the active state of managed devices and systems. One can also consider Sysman as a vehicle for providing named pipes [6] where one side of the pipe is connected to a remote device. Sysman is an easily extensible framework. Sysman allows the inclusion of new devices by simply allowing agents and scripts developed for these devices be utilized by Sysman. Here are the contributions of Sysman:

- Sysman simplifies management of HPC clusters and data centers
- Provides a unified interface for managing different system types, servers, storage systems and network devices.
- File system representation of managed systems enables the use of simple and familiar, yet powerful UNIX (and Linux) file system commands for managing hundreds and thousands of systems as easily as a single system
- Sysman can be extended to support arbitrary device types provided that device operations can be represented as file read and write operations
- Since Sysman and its agents run in user-space, developing new extensions is easy.
- Graphical and web based interfaces can be built on top of Sysman

The rest of this paper includes a background followed by the basic architecture of Sysman and advanced design issues. Related work, future work, and conclusions complete the presentation.

Background

The *proc* file system [18] is a virtual file system used in UNIX systems. Information about the system can be obtained by accessing files in the /proc directory. Linux, SUN Solaris, and IBM AIX are among operating systems which support such a virtual file system. Certain system parameters can be configured by writing to files in the /proc directory. The /proc file system, and the similar /sys file system are used for obtaining information and configuring local resources only. Sysman operates on a similar basis, but as opposed to the /proc system, it is used for managing networked systems. Therefore, it can be used to configure a much more diverse group of systems.

Sysman is a virtual file system developed completely in user space through the use of FUSE, *File system in User-space* [4]. FUSE, is a kernel module, which provides a bridge to the kernel interface. FUSE is now part of the Linux kernel. It provides a simple API, has a very efficient user-space kernel interface and can be used by non privileged users. The path of a typical file system call is shown in Figure 2a. FUSE can intercept file system calls and redirect them to a user-space program for implementing virtual file systems. Several virtual file systems have been developed on top of FUSE. These file systems provide various features, from versioning, to encryption, to simple methods for accessing special devices. A list of file systems built on top of FUSE can be found at [3].

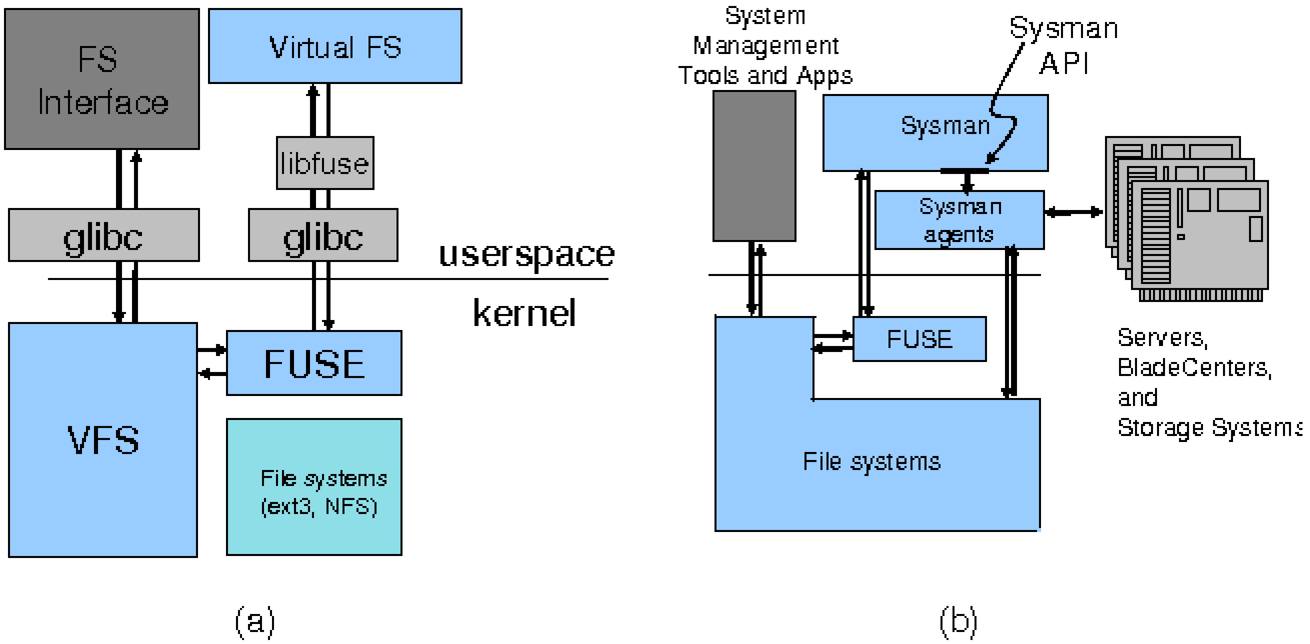

Figure 2: (a) File system call with FUSE and (b) path of typical system administrative tasks with Sysman.

In Sysman, we use FUSE to intercept accesses to Sysman files. While accesses to other files are not affected, certain access types to Sysman files result in information gathering, configuration, and other system administrative tasks. In particular, Sysman intercepts accesses to its directories. In the next section we provide discuss the basic architecture of Sysman.

Basic Architecture

In Sysman, each device is represented as a separate directory with its own files and possibly subdirectories under the /sysman subtree. Figure 2b illustrates the path of typical system administrative tasks. When as part of a system administration task, a file in the /sysman directory is accessed, the file system call is intercepted by FUSE and its processing gets delegated to the user level Sysman program. In the rest of this section we discuss various aspects of Sysman in detail.

Virtual File System Operations

Main system management tasks are performed by either reading from a file in the /sysman directory or by writing into a file in this directory. When a file in /sysman directory is accessed, Sysman determines if there is an agent associated with the requested <file name, file system operation> pair, and executes the matching agent, if it exists, before processing the access as a regular file system read or write. For each <file name, file system operation> pair, the name of the corresponding agent is derived by Sysman. For example, discover_master.write is the agent to execute on write operations to the file discover_master. Sysman stores the agents in a hierarchical manner in a configurable location and therefore given the complete path of a file, the corresponding agent name and location can be easily found. If such an agent exists, it is executed. If not, no agents will be executed. In either case, the file access operation completes as a regular file system operation. That is, if the file system operation is a read from a file, Sysman reads the content of the file, or if the file system operation is a write to a file, it writes into the file.

In general, read operations are used for obtaining information about the status of devices that are being managed. For example, by reading the content of the file *power* in a server directory, one can find out if the server is turned on or not. Sysman supports various information caching models that affect how often system management information is collected. Whenever a file is read, if the information is not already stored in the file, necessary action is taken to obtain the information, store it in the file and then present it to the user. If the information is already available, the behavior depends on the caching model used for that particular file. The cached information can be simply presented to the user or the content can be refreshed first. In certain cases, the information is collected periodically. The caching model is specified in the Sysman configuration file and can be set for a group of files or individual files. We will discuss this feature in more detail in the Special Files and Error Handling section.

On the other hand, writes to certain files result in performing a task on the corresponding device (or component). For example, writing a 1 or the word *on* into a *power* file results in Sysman turning on the server. In cases where the result of the operation is required to be preserved, the result is stored in the file. A subsequent read from the file prints out the result in those cases. Sysman also recognizes certain file and directory names and extensions and performs certain predefined actions in response to accessing them. This feature is discussed in the Special Files and Error Handling section.

The /sysman Directory Hierarchy

The /sysman directory is organized such that each managed device has its own directory. Furthermore, similar devices (e.g., servers with the same type, or blades of an IBM BladeCenter) are listed under one directory. Files in each directory are used for managing the device represented by that directory. Similarly files in parent directories are used for obtaining information about the group a device belongs to. Figure 3 shows parts (not all) of the /sysman directory structure for a BladeCenter system in our lab consisting of eight chassis with 14 blade servers in each chassis (112 blade servers total) and several other servers. The creation of this directory tree is discussed later when we discuss the discovery mechanisms. Note the 8 chassis found under the directory /sysman/systems/chassis/. Each chassis is represented by a directory named after the IP address of the chassis management module, for example 10.10.1.1.

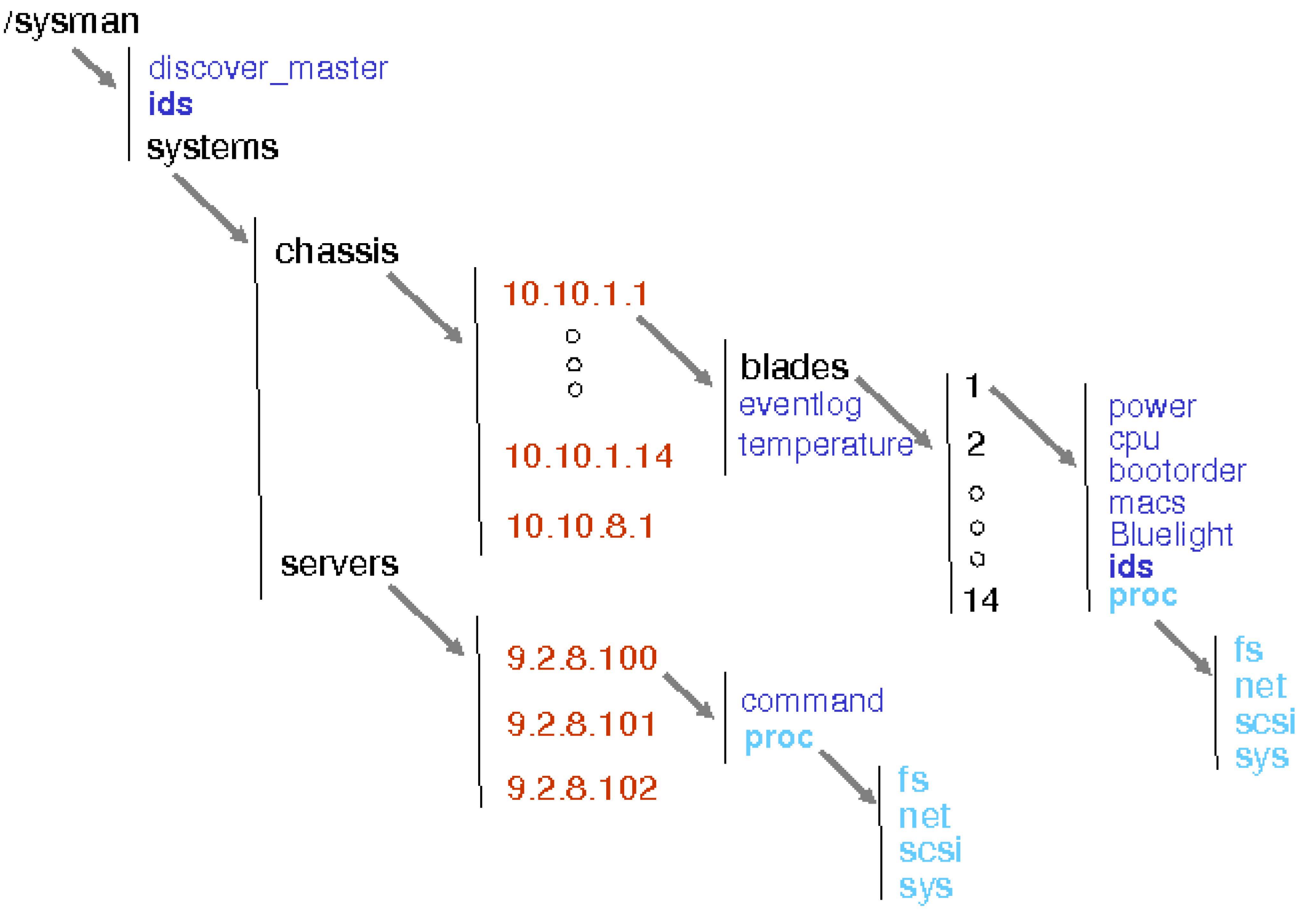

Figure 3: Partial snapshot of a /sysman directory.

In Figure 3, under each chassis subdirectory there is a *blades* directory and two files, *temperature_mp* and *eventlog*, which report the hardware temperature and hardware events common to the chassis, respectively. Blades directory contains up to 14 subdirectories named 1 to 14, each representing a blade server found in the chassis. Reading from files found under the blade directory, reports the blade status and properties. For example, the first command in Figure 4 displays the boot order of blade number 3 in the BladeCenter chassis whose Management Module (MM) IP address is 10.10.1.1. Likewise, for configuring devices, Sysman files may be written to. For example, to change the boot order of the blade, one may execute the second command in Figure 4.

```
$ cat /sysman/system/chassis/10.10.1.1/blades/3/bootorder
$ echo "network, floppy, cdrom, harddisk" > bootorder
```

Figure 4: Examples of reading from and writing to Sysman files.

Sysman is configured by a single configuration file. This configuration file specifies among other things the location of the Sysman agents (Figure 5a) and where the FUSE directory is mounted. By default, any access to a Sysman file refreshes the content of the file first. On the other hand, by default accessing a directory does not cause refreshing the directory except for *proc* and *sys* directories.

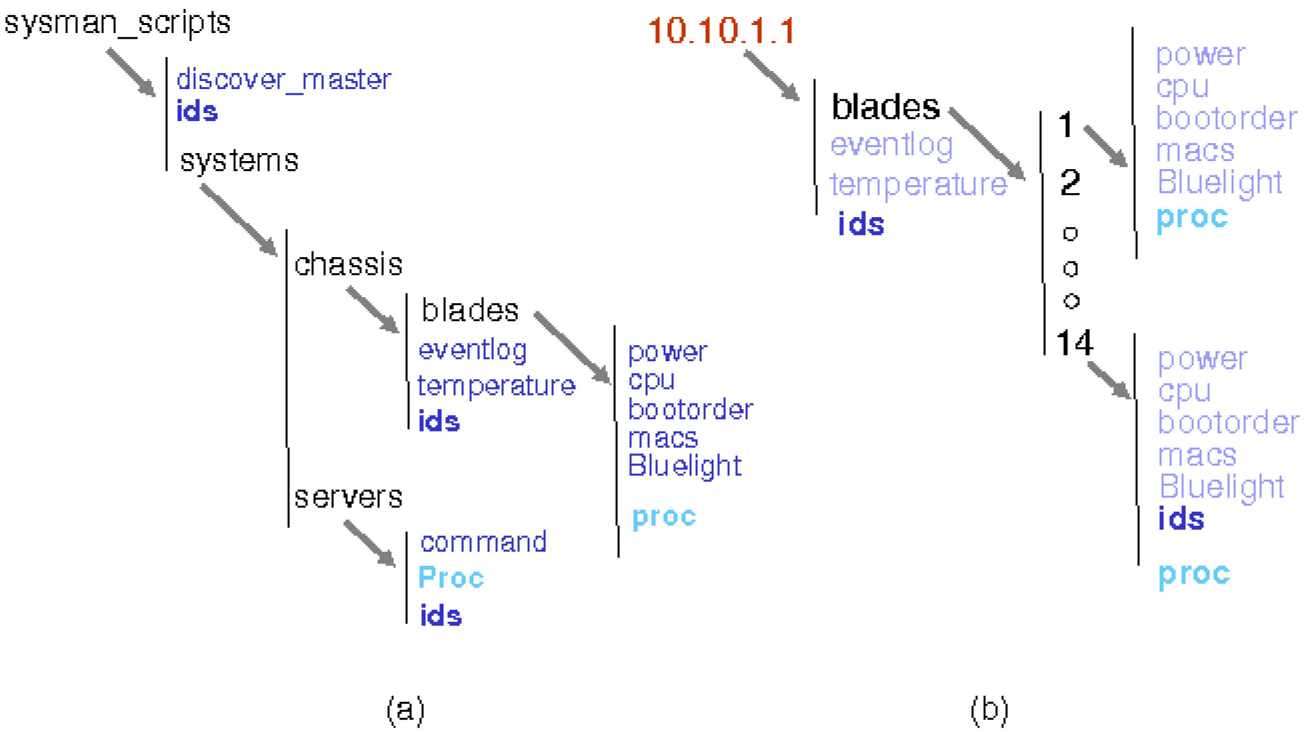

Figure 5: (a) Parts of a Sysman script directory and (b) different ids files.

Discovery

Sysman provides three methods for adding new devices to the system: prompted, automatic, and manual.

The prompted method requires the writing of `1' to the /sysman/discover_master file. Once the file is written to, Sysman searches for devices known to it and appropriate directories and files are created upon discovery of new devices. BladeCenter chassis are discovered through the Service Location Protocol (SLP) [25]. The prompted method is used by issuing a command like the first command in Figure 6. This command causes Sysman to run the discover_master.write script which runs an SLP client program. The SLP client makes broadcast announcements to discover all the BladeCenter chassis on the local subnet. For each chassis discovered, a directory is created in the /sysman/system/chassis directory, with the IP address of the chassis management module as the name of the directory.

```
$ echo "1" > /sysman/discover_master
$ echo "server 10.10.2.15" > /sysman/discover_master
```

Figure 6: Methods of discovery in Sysman.

In the automatic method, Sysman periodically runs the same discovery processes as described above without user intervention. A discovery frequency is specified in the configuration file. When a new device is discovered that contains other devices, discovery is run automatically on the new device as well. In the manual method, the name or IP address of the device (or its management interface), along with the device type is written to the discover_master file. As a result of such a write, corresponding directories and files are created in the /sysman directory. The second command in Figure 6 adds a server to the system.

Sysman File System Security

Sysman uses the existing file system authentication and permission system used in Linux to control access to the files under /sysman. Access to /sysman can be limited to the root user if need be. If the access is not limited to root, regular file system permissions are checked to see if a user can access a file under /sysman. For example, users in a system administration group can be given both read and write access to sysman files, while other users can be limited to read access. In another example, a user or a group can be restricted to accessing only a subtree of /sysman, therefore authorizing those users to manage only a subset of the managed systems. Furthermore, other available access control lists can be also used for enhanced file security.

The Command File

A generic method for running tasks on a remote server using ssh is provided by writing the command into the *command* file. When a command is written to a file called *command* in the /sysman directory tree, the Sysman executes that command remotely. The result of the remote execution is written into the *command* file. A subsequent read from this file prints out the result of the command execution. The command agent provides a simple method for Sysman users to perform tasks not found under /sysman. This is similar to the C function system(<*string*>), except that the <*string*> runs on multiple systems found under the /sysman tree. For example, to get a list of running processes from all systems, one could execute the sequence of commands in Figure 7.

```
$ echo "ps -ax" > run_this
$ find /sysman/systems/servers -name command -exec cp run_this {} \;
$ find /sysman/systems/servers -name command -exec cat {} \;
```

Figure 7: Issuing commands to Sysman devices.

Advanced Features

In this section we describe some of the more advanced features of Sysman.

Authentication for Access to Managed Systems

Many managed systems and devices require their own authentication method or information. Since Sysman wraps other system management tools, Sysman must support a way to comply with the authentication requirements of the lower level tools and cache credentials. This is done through the use of the *ids* files. The user id and password are provided by the user by writing them into the ids file. These values are then encrypted and kept by Sysman in memory and are **not** written to a file as a security measure. Sysman agents can look up these credentials whenever they need them.

Inheritance and Special Files

Sysman supports inheritance in the /sysman hierarchy. One example of this feature is the *ids* file which contains <userid,password> pairs. In Figure 5b the user id and password for accessing blade 14 is found in the *ids* file under the blade 14 directory. However, blade 1 does not have its own *ids* file. In Sysman, it is not necessary to create an *ids* file for each managed device or system. The user ids and passwords propagate down directory trees for the convenience of the user.

If a system management operation requires a user id and password, but there is no *ids* file in the corresponding directory, the ids of parent directory is searched for such an entry. This process is continued up in the directory tree until an ids file is found. Then the user id and password are retrieved by accessing the *ids* file. (The /sysman/ids file represents the top level *ids* file.)

Special Files and Error Handling

Sysman recognizes some files such as *ids* files as special files. Accesses to special files leads to the execution of predefined tasks by Sysman. Another special file is the *lock* file. In cases where a resource is shared by multiple devices (for example the management controller of several blades) and access to the shared resource needs to be serialized, Sysman provides the required mechanism for locking (by using a file called *lock* in the location corresponding to the shared resource of interest).

Special files are used for handling errors. In Sysman, files ending with *.status* extension are special files. These files follow a simple format:

```
<return_value [return_message]>
```

Sysman and its agents use this file to store the return value of an operation (an integer) and possibly a human readable message indicating what the result is. If the operation succeed the return value will be zero. For example the status of executing a command initiated by writing to the *power* file in a directory will be stored in *power.status* in the same directory.

This feature can be used to implement various error handling methods at Sysman level. That is, Sysman users can write their programs and scripts such that depending on the result of an operation, suitable actions can be taken. Another level of error handling can be performed at the level of agents. Agents used for managing devices can do more than passing the result status by using *.status* files. Agents can themselves be responsive to various error conditions and try to recover from various failures. (Our current implementation uses agents which can respond to failing connections, etc.)

Unified /proc and /sys

Directories called *proc* and *sys* are special directories in Sysman. Sysman consolidates information provided under the /proc and /sys directories of all managed Linux systems. Since these directories change dynamically, their content gets updated by accessing the corresponding remote device, whenever they are accessed. In other words, by default these directories will be refreshed upon access. Additional directories which require the same treatment can be defined in the Sysman configuration file.

API and Extensibility

Sysman supports a simple extensible interface for calling its agents (the underlying scripts and binary executables). Whenever a Sysman file is accessed the name and path of the agent to be executed are derived by Sysman. Then the agent is run and a set of parameters are passed to the agent as options (Figure 8). Currently, in addition to the complete path and file name of the file being accessed, three options are supported. User id and password (the *-u* and *-p* options) are used when accessing the device requires the use of a user id and password. The *-i* option is used to pass the information provided by user for accessing the file. For example, the content the user wants to write into a file is passed to the agent through the use of this option. As it can be seen, this interface is very simple and flexible. Agents for performing new tasks and/or supporting new devices can be easily developed by following this interface. Once the agent is developed and tested, it can be easily added to the Sysman directory for agents.

```
agent -f file_name [-u user_id]
        [-p password] [-i user_input]
```

Figure 8: Sysman API.

Asynchronous Execution

The main strength of Sysman is that it can be used along with a variety of file system and operating system commands and utilities. Sysman can be utilized in an asynchronous manner simply by accessing Sysman files asynchronously (i.e., in the background). This feature is very useful when hundreds and thousands of devices are being managed by Sysman. Let us consider a case where we want to obtain the CPU utilization (and other information provided by the *vmstat* command) of our compute nodes. If we collect this information node by node it will take a long time before all nodes are contacted. Alternatively by accessing corresponding files in the background as shown in the example below by using the ``&" sign we can initiate access to the nodes without waiting for the responses to arrive (Figure 9).

```
## A simple script to collect the vmstat
## of all servers in background
#!/bin/bash
LIST=`find /sysman/systems -name command`
for i in $LIST; do
  echo "vmstat " > $i &
done
```

Figure 9: Asynchronous execution of Sysman commands.

Figure 10 shows how long it takes to get this information from up to more than 2000 nodes synchronously and asynchronously. It can be observed that when the operation is done asynchronously, the parallelism in obtaining the information from different nodes results in an order of magnitude reduction in total time.

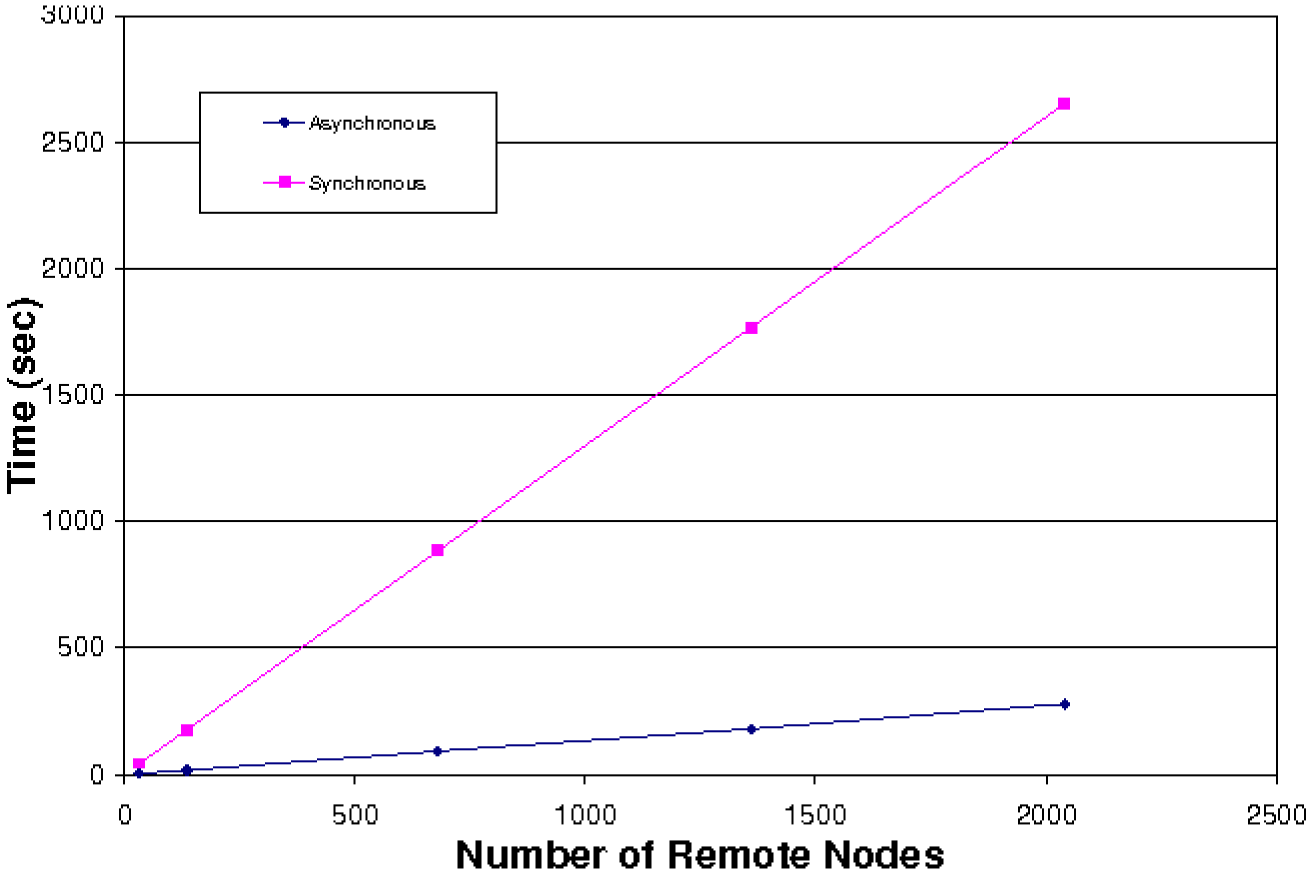

Figure 10: Sysman synchronous and asynchronous execution modes.

Related Work

The difficulties in managing clusters containing many physical machines and now hundreds and thousands of virtual machines have led several research and development groups to work on various methods for automating system administrative tasks. Very different approaches have been taken by these groups. In this section we discuss some of the related work.

Plan 9 is a distributed system built at AT&T Bell Laboratories [24] in which all system resources are represented by a hierarchical file system. Files in this file systems are not repositories for storing data on disk. However, resources are accessed using file-oriented read and write calls. The network protocol used for accessing and manipulating system resources is called 9P. Sysman uses a similar method for performing system management tasks.

Xcpu [22] is a 9P based framework for process management in clusters and grids developed as Los Alamos National Laboratory intended to be a replacement for the bproc system for managing processes across a cluster. The system uses 9p to develop a directory tree to control the scheduling and execution of jobs on a large cluster. Each node in the system is represented by a directory, including subdirectories for all sessions running on the nodes. All processes are run within a session.

The session directory includes the files arch, ctl, exec, and status among others. The arch file can be read indicating the architecture of the node, and the status file reports on the status of the session. The ctl and exec files are used to control the execution of a process in the session. The target image to run on the node is specified by copying the image to the exec file, while the control of the session is controlled by writing various values to the ctl file.

Xcpu uses a similar architecture in representing system resources as directories and files that can be accessed for system management tasks. Configfs [2] is a user-space driven approach for configuring kernel objects. Similar to Xcpu and Sysman, resources are presented in a directory hierarchy and configuration tasks are achieved by accessing files. While Xcpu is mainly used for process management and Configfs is used for configuring kernel objects, Sysman is used for managing a larger and more diverse set of resources.

Nagios [8] is a host and network monitoring application for monitoring *private* services and attributes such as, memory usage, CPU load, disk usage, and Running processes. Nagios watches hosts and services specified to it, and alert the user when things go bad and when they get better. It provides the ability to define event handlers to be run during service or host events. Ganglia [5] is another distributed monitoring system for high-performance computing systems such as clusters and Grids. It uses It uses widely used technologies such as XML, XDR, and RRDtool for data storage and visualization. IBM Cluster Systems Management (CSM) [1] is a management tool for clustered and distributed Intel and PowerPC systems. CSM supports Linux and AIX operating systems and can be used for system deployment, monitoring and management. In particular, CSM is designed to provide the parallelism required for efficient management of clusters made of hundreds of nodes. In contrast with these tools and products, for system administrators familiar with UNIX and UNIX-like environments, Sysman is very simple and can be deployed and used effectively in a very short amount of time.

Usher [21] and MLN [15] aim for providing an extensible framework for managing large clusters and networks. In particular, Usher is a virtual machine management system with an interface whereby system administrators can ask for virtual machines and delegate the corresponding administrative tasks to modular plug-ins. The architecture of Usher is such that new plug-ins can be developed and utilized by users. The goals behind the architecture of Sysman is similar to those of Usher in that they both aim to provide an extensible framework where users can add their own modules (scripts) when need be. Sysman differs from Usher in its scope and the fact that it tries to target wider array of systems (including virtual systems) and devices such as storage and network devices.

Future Work and Conclusions

An early version of Sysman with limited set of features has been available on SourceForge [9]. We are extending our work to support storage devices as well. Currently we have developed several agents for managing IBM DS4000 series storage systems. These agents are used to create LUNs and map them to appropriate host systems. Although not discussed in the current version of the paper, we have completed this work. We are also extending Sysman to monitor and manage networking devices (in addition to those network switches which can be managed through IBM BladeCenter management modules and are supported).

We are investigating scalable methods for supporting simultaneous and asynchronous execution of multiple instances of agents. We plan to work on providing a more robust error handling mechanism. We also plan to extend Sysman to cover the management of virtual machines as well. Using the interface for managing Linux KVM [7], Xen [17] and VMware [11] virtual machines, Sysman will be used to create and manage virtual machines. This will include the management of storage volumes for these virtual machines as well. Another direction for future is to enhance Sysman to become a distributed management system in which multiple Sysman servers cooperate with each other to provide a higher level of scalability and fault tolerance. We are investigating the use of multicast messages for issuing the same commands to multiple systems too.

In this paper, we presented Sysman, a new infrastructure for system management. Sysman provides a simple yet very powerful interface which makes the automation of system management tasks very easy. Since Sysman is presented to end users as a virtual file system, all UNIX file system commands can be utilized to manage a cluster of possibly heterogeneous servers and other devices. Sysman is designed to be easily extensible. Agents for new devices can be developed and added to the system easily. Furthermore, graphical and web based interfaces can be added on top of Sysman.

Author Biographies

Mohammad Banikazemi is a research staff member at IBM T. J. Watson Research Center. His research interests include computer architecture, storage systems and simplifying server and storage management. He has a Ph.D. in Computer Science from Ohio State University, where he was an IBM Graduate Fellow. He has received an IBM Research Division Award and several IBM Invention Achievement Awards. He is a Senior member of both the IEEE and the ACM.

David Daly is a research staff member at IBM T. J. Watson Research Center. He received a B.S. degree in computer engineering in 1998 from Syracuse University. He received Ph.D. and M.S. degrees in Electrical Engineering in 2005 and 2001 respectively, both from the University of Illinois at Urbana-Champaign. Since joining IBM in 2005, he has worked on computer system performance analysis and modeling, analyzing financial workloads for high performance computing, and simplifying server management.

Bulent Abali received his Ph.D. from the Ohio State University in 1989. He has been a research staff member at IBM Research since then.

Bibliography

[1] *Cluster Systems Management (CSM)*, https://www.ibm.com/systems/clusters/software/csm/index.html.

[2] *Configfs*, https://www.mjmwired.net/kernel/Documentation/filesystems/configfs/.
[3] *File Systems Using FUSE*, https://fuse.sourceforge.net/wiki/index.php/FileSystems.
[4] *FUSE: File System in Userspace*, https://fuse.sourceforge.net/.
[5] *Ganglia*, https://ganglia.sourceforge.net/.
[6] *Introduction to Named Pipes*, https://www.linuxjournal.com/article/2156.
[7] *Kernel Based Virtual Machine*, https://kvm.qumranet.com/kvmwiki.
[8] *Nagios*, https://www.nagios.org/.
[9] *Sysmanfs: A Virtual Filesystem for System Management*, https://sourceforge.net/projects/sysmanfs.
[10] *The openMosix Project* https://openmosix.sourceforge.net/.
[11] *VMware*, https://www.VMware.com.
[12] Amar, L., A. Barak, E. Levy, and M. Okun, ``An On-Line Algorithm For Fair-Share Node Allocations in a Cluster," *Proceedings of Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid'07)*, 2007.
[13] Amar, L., J. Stoesser, A. Barak, and D. Neumann, ``Economically Enhanced Mosix for Market-based Scheduling in Grid OS," *Proceedings of Workshop on Economic Models and Algorithms for Grid Systems (EMAGS) 2007, 8th IEEE/ACM International Conference on Grid Computing (Grid 2007)*, 2007.
[14] Barrett, R., E. Kandogan, P. P. Maglio, E. Haber, L. A. Takayama, and M. Prabaker, ``Field Studies of Computer System Administrators: Analysis of System Management Tools and Practices," *Proceedings of ACM Conference on Computer Supported Cooperative Work*, 2004.
[15] Begnum, K., ``Managing Large Networks of Virtual Machines," *LISA '06: Proceedings of the 20th Conference on Large Installation System Administration*, p. 16, 2006.
[16] Daly, D., J. H. Choi, J. E. Moreira, and A. P. Waterland, ``Base Operating System Provisioning and Bringup for a Commercial Supercomputer," *Proceedings of The Third International Workshop on System Management Techniques, Processes and Services (SMTPS)*, 2007.
[17] Dragovic, B., K. Fraser, S. Hand, T. Harris, A. Ho, I. Pratt, A. Warfield, P. Barham, and R. Neugebauer, ``Xen and the Art of Virtualization," *Proceedings of the ACM Symposium on Operating Systems Principles*, 2003.
[18] Faulkner, R. and R. Gomes, ``The Process File System and Process Model in UNIX System V," *USENIX Winter*, pp. 243-252, 1991.
[19] *IBM BladeCenter*, https://www.ibm.com/systems/bladecenter/.
[20] Massie, M. L., B. N. Chun, and D. E. Culler, ``The Ganglia Distributed Monitoring System: Design, Implementation, and Experience," *Parallel Computing*, Vol. 30, Num. 7, 2004.
[21] McNett, M., D. Gupta, A. Vahdat, and G. M. Voelker, ``Usher: An Extensible Framework for Managing Clusters of Virtual Machines," *Proceedings of the 21st Large Installation System Administration Conference (LISA)*, November, 2007.
[22] Minnich, R. and A. Mirtchovski, ``Xcpu: A New, 9p-Based Process Management System for Clusters and Grids," *Proceedings of IEEE International Conference on Cluster Computing*, 2006.
[23] Mirtchovski, A. and L. Ionkov, ``Kvmfs: Virtual Machine Partitioning for Clusters and Grids," *Proceedings of The Linux Symposium*, 2007.
[24] Pike, R., D. Presotto, K. Thompson, H. Trickey, and P. Winterbottom, ``The Use of Name Spaces in Plan 9," *Operating Systems Review*, Vol. 27, Num. 2, 1993.
[25] *RFC 2608, Service Location Protocol (SLP), Version 2*, https://tools.ietf.org/html/rfc2608.
[26] Sacerdoti, F., M. Katz, M. Massie, and D. Culler, ``Wide Area Cluster Monitoring with Ganglia," *Proceedings of IEEE International Conference on Cluster Computing*, 2003.